40195

National Library of Canada | Bibliothèque nationale du Canada

CANADIAN THESES ON MICROFICHE | THÈSES CANADIENNES SUR MICROFICHE

NAME OF AUTHOR/*NOM DE L'AUTEUR* Paul Cheuk Cheung Kam

TITLE OF THESIS/*TITRE DE LA THÈSE* Retrieval in Clustered file

UNIVERSITY/*UNIVERSITÉ* University of Alberta

DEGREE FOR WHICH THESIS WAS PRESENTED/
GRADE POUR LEQUEL CETTE THÈSE FUT PRÉSENTÉE M. Sc.

YEAR THIS DEGREE CONFERRED/*ANNÉE D'OBTENTION DE CE GRADE* 1978

NAME OF SUPERVISOR/*NOM DU DIRECTEUR DE THÈSE* Dr. Clement Yu

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

DATED/*DATÉ* 28-6-78 SIGNED/*SIGNÉ* Paul Kam

PERMANENT ADDRESS/*RÉSIDENCE FIXE* Box 110 U. of A.
Edmonton, Alberta
T6G 2E0

NL-91 (3-74)

National Library of Canada

Cataloguing Branch
Canadian Theses Division

Ottawa, Canada
K1A 0N4

Bibliothèque nationale du Canada

Direction du catalogage
Division des thèses canadiennes

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

THE UNIVERSITY OF ALBERTA

RETRIEVAL IN CLUSTERED FILE

by

© PAUL CHEUK CHEUNG KAM

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

IN

COMPUTING SCIENCE

DEPAPTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

FALL, 1978

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled RETRIEVAL IN CLUSTERED FILE submitted by PAUL CHEUK CHEUNG KAM in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE in COMPUTING SCIENCE.

.....
Supervisor

.....

.....

Date. 27th June 1978

TO MY PARENTS

ABSTRACT

The clustered file organization and the search algorithms associated with it are studied in detail. A method of estimating the number of records which are 'closest' to a given query in a clustered file organization is implemented. Since the data base used to test the estimation does not satisfy exactly the assumption of the model, a higher error percentage than expected is obtained from the result and thus a modification scheme is developed to improve the accuracy of the estimation.

ACKNOWLEDGEMENTS

This author is indebted to his supervisor Dr. C. T. Yu for his guidance and support throughout the course of the study.

I would like to thank my examination committee Prof. W. S. Adams and Dr. T. O. Maguire for their advices and criticisms.

Gratitude is also extended to Raymond Lacousta for drawing the diagrams and mathematical formulae and John Demco for proof reading the thesis.

TABLE OF CONTENTS

PAGE

CHAPTER 1. INTRODUCTION

1.1 INFORMATION RETRIEVAL SYSTEMS.....1

1.2 FILE ORGANIZATION.....5

1.3 RETRIEVAL IN CLUSTER FILE.....16

CHAPTER 2. CLUSTER FILE RETRIEVAL SYSTEMS

2.1 A PROBABILISTIC MODEL.....18

2.2 TO COMPUTE THE C FUNCTION.....22

2.3 EXPERIMENT ON A REAL DATA SET.....25

CHAPTER 3. ERROR ANALYSIS AND QUERY MODIFICATION

3.1 ERROR ANALYSIS.....31

3.2 ASSUMPTION RELAXATION.....32

3.3 THE DEPENDENT ATTRIBUTES.....35

3.4 EXPERIMENT SETUP AND RESULT.....37

CHAPTER 4. TIME BOUND ANALYSIS OF ALGORITHMS

4.1 INTRODUCTION.....46

4.2 TIME BOUND TO COMPUTE C-FUNCTION ACCORDING TO ITS DEFINITION.....48

4.3 TIME BOUND FOR POLYNOMIAL MULTIPLICATIONS.....50

REFERENCES.....54

TABLE OF CONTENTS CONTINUED

PAGE

APPENDIX I.....57

APPENDIX II.....58

APPENDIX III.....59

CHAPTER 1

INTRODUCTION

1.1 INFORMATION RETRIEVAL SYSTEMS

Digital computers were originally designed for solving mathematical problems and the like. However with the constant demand of the public and the introduction of direct-access mass storage and sophisticated operating systems, it was transformed into a means to process information. Since then hundreds of information storage and retrieval systems have been developed. Some of them were multi-purpose, while the others were aiming at some specific functions.

Basically an information retrieval system is composed of a data base, a retrieval subsystem and a data base maintainence subsystem. The retrieval subsystem is used to analyse the user queries and retrieve the appropriate information. The data base maintainence subsystem is used to keep the data base up-to-date by performing such functions as the addition of new records, the deletion of obsolete records and the modification of existing records.

In general, according to the organization of the data base and the way that the information is retrieved, information retrieval systems can be broadly classified into two groups: Data Retrieval Systems and Document Retrieval

Systems.

The records of a data retrieval system are usually composed of a fixed number of fields or attributes. Each record is completely characterized by the values of its attributes and is uniquely identified by the value of a key field or attribute. For example, in a student record retrieval system, each record may consist of 4 attributes: student number, name, age and sex. Each record is uniquely identified by the value of the student number. The retrieved information has to satisfy exactly the conditions specified by the user queries. For example, the retrieved records to the query

find AGE < 20 AND SEX = female

represent the group of female students who are under 20 years of age.

Data retrieval systems are widely used by the general public, especially in the commercial sector, in which management decision problems as well as routine inventory problems have to be solved.

Document retrieval systems, on the other hand, deal with items which may not be completely characterized by the values of a fixed set of attributes. Example of items are books and pictures. A set of descriptors or keywords is used to represent and describe the content of the items instead. The choice of the keywords or descriptors is quite

subjective. Usually the contents of an item are translated into a record of keywords by using a manually prepared dictionary. User queries are formed in the same way (by using the same dictionary). Queries are matched or correlated with the records within the data base and the records which have a substantial number of keywords in common with the queries are retrieved. Note that the retrieved records do not have to match the query exactly as in the case of the data retrieval system.

In a data retrieval system, since all the records are completely characterized by a fixed number of attributes, the retrieved records are always acceptable to the users as long as they satisfy the conditions specified by the user query. However this is not the case in a document retrieval system. There are some factors we have to consider regarding the relevancy of the retrieved information. First, in forming a query, the user may not be able to choose the most appropriate keywords, thus the retrieved records may turn out to be different from those the user is expecting. Second, the keywords which were chosen by the indexer may have different meanings to the users. Third, the matching function used may not accurately measure the closeness between the query and the records. Thus we would have a situation such that some of the closest records as measured by the matching function using the supplied keywords may not be considered relevant while some non-retrieved records may be of interest to the users.

One approach to alleviate the above problems in an on-line processing environment is by employing an iterative process known as relevance feed back. Relevance feed back is a process such that the users, by responding to the displayed information, identify the relevant and irrelevant documents from the retrieved documents and have this information fed back to the system. The system then alters the queries by adding heavier weights to the keywords of the relevant documents and reducing the importance of the attributes found in the irrelevant documents. A new search is then made and the result is presented to the user and the entire process is repeated until the user is satisfied with the result. One of the retrieval systems which employs the relevance feed back technique is the SMART retrieval system[1].

We can conceive that an on-line processing environment is quite suitable for a document retrieval system and it is useful to investigate some aspects involved. In particular, one important aspect that the success of an on-line processing environment might depend upon is the file organization. In the next section, several file organizations which may be suitable for on-line processing are discussed.

1.2 FILE ORGANIZATION

The different objectives for different information systems demand the development of various data or file organizations. Each type of objective may call for a different file organization which will better satisfy the requirements. In some cases, such as in an inquiry system, data must be retrieved rapidly while updating of existing data can proceed at a slower rate. In other cases, such as an order entry system, it is necessary to store rapidly a large volume of data which are to be retrieved at a slower pace.

Since this thesis is interested in document retrieval systems and in particular the file organizations, it is important at this point to examine several file organizations which support document retrieval systems. In all, four types of file orgnization in a document retrieval system environment are discussed: sequential, chained, inverted and clustered.

A. Sequential Files

Records or documents in a sequential file are stored in the order of acquisition and the ith record is retrieved by a sequential scan of the previous i-1 records. The access

time in a sequential file is so large that this kind of file organization is normally not suitable for on-line retrieval. Furthermore the sequential scheme also presents problems in updating and deleting of records. In particular, after a deletion, unless the file is reorganized, the unused space once occupied by the deleted record is rather difficult to recover or reuse.

In spite of the above disadvantages, sequential files have some favorable properties. Since the entire record can be examined directly, any information about the documents can be retrieved at once. Furthermore the overhead is low, since no pointers or directory are involved. The sequential file organization is suitable for a data acquisition system or a system which does not involve too much retrieval but has to produce many reports from the existing records.

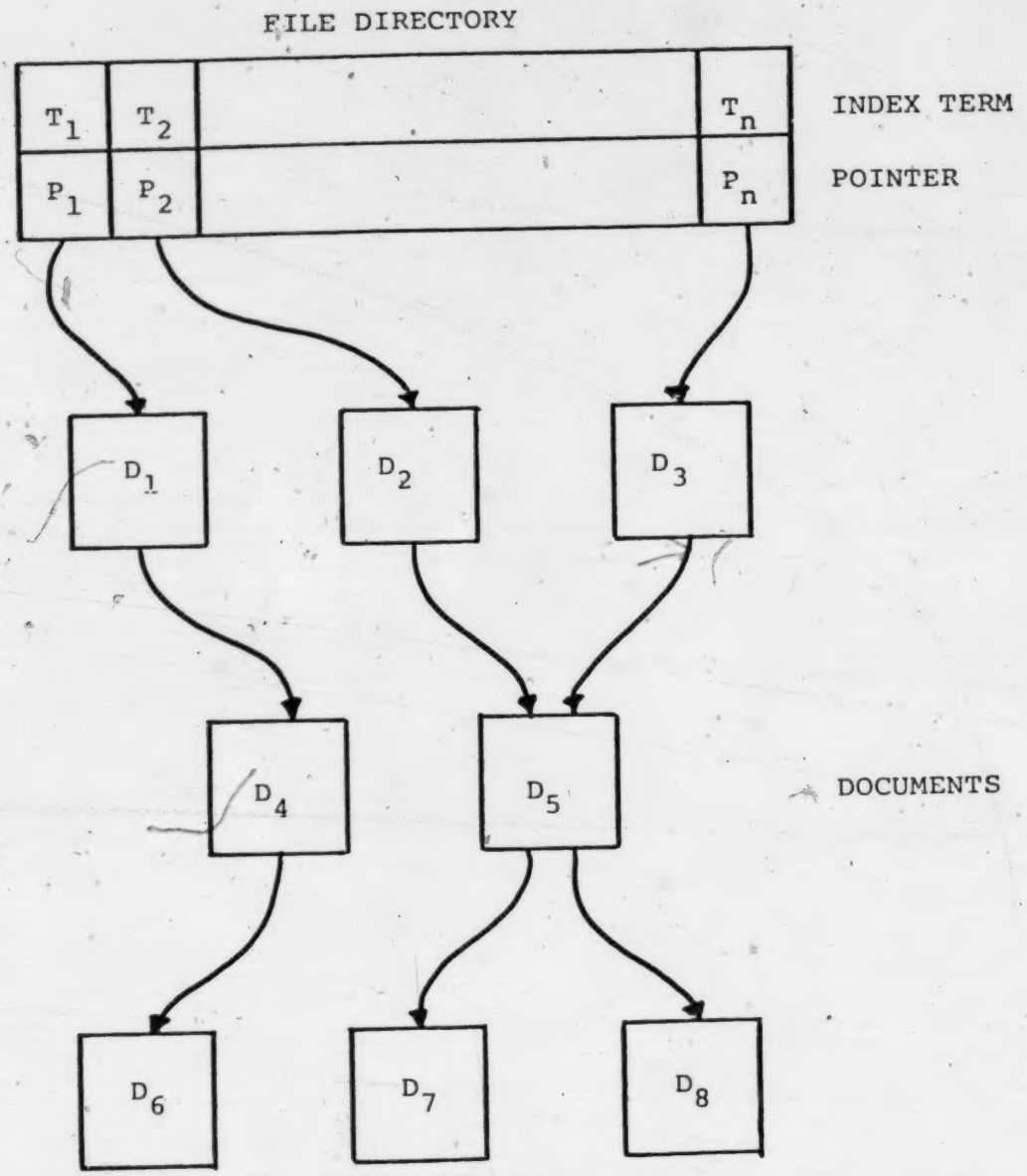
B. Chained Files

Documents or records having common terms or keywords are grouped together to form a set. The documents in each set are chained together by using pointers. A file directory is used to facilitate the searching of chain heads. The file organizations is depicted in figure 1.1. The head of each chain can be located by using binary search, hashing or some other table look up techniques. Subsequently, elements of all the chains are fetched and correlated with the query. The ones with 'high' correlations are retrieved.

Owing to the structure of the chained file, a number of

pre-search statistics are available. For example, the total length of all the chains to be retrieved can be used as an upper bound to the number of retrievable documents. Furthermore, the length of a particular chain may also be of interest to some people.

Despite the features mentioned above, there are some drawbacks in using the chained file. First, a great deal of work has to be done in order to update or delete a document. Second, the number of documents with many terms in common with the query is usually small as compared to the total number of documents accessed. Third, the heavy storage overhead for the directory and pointers may make the system too costly to implement. Finally, the merging of the actual documents from each chain deteriorates the retrieval time to a great extent.

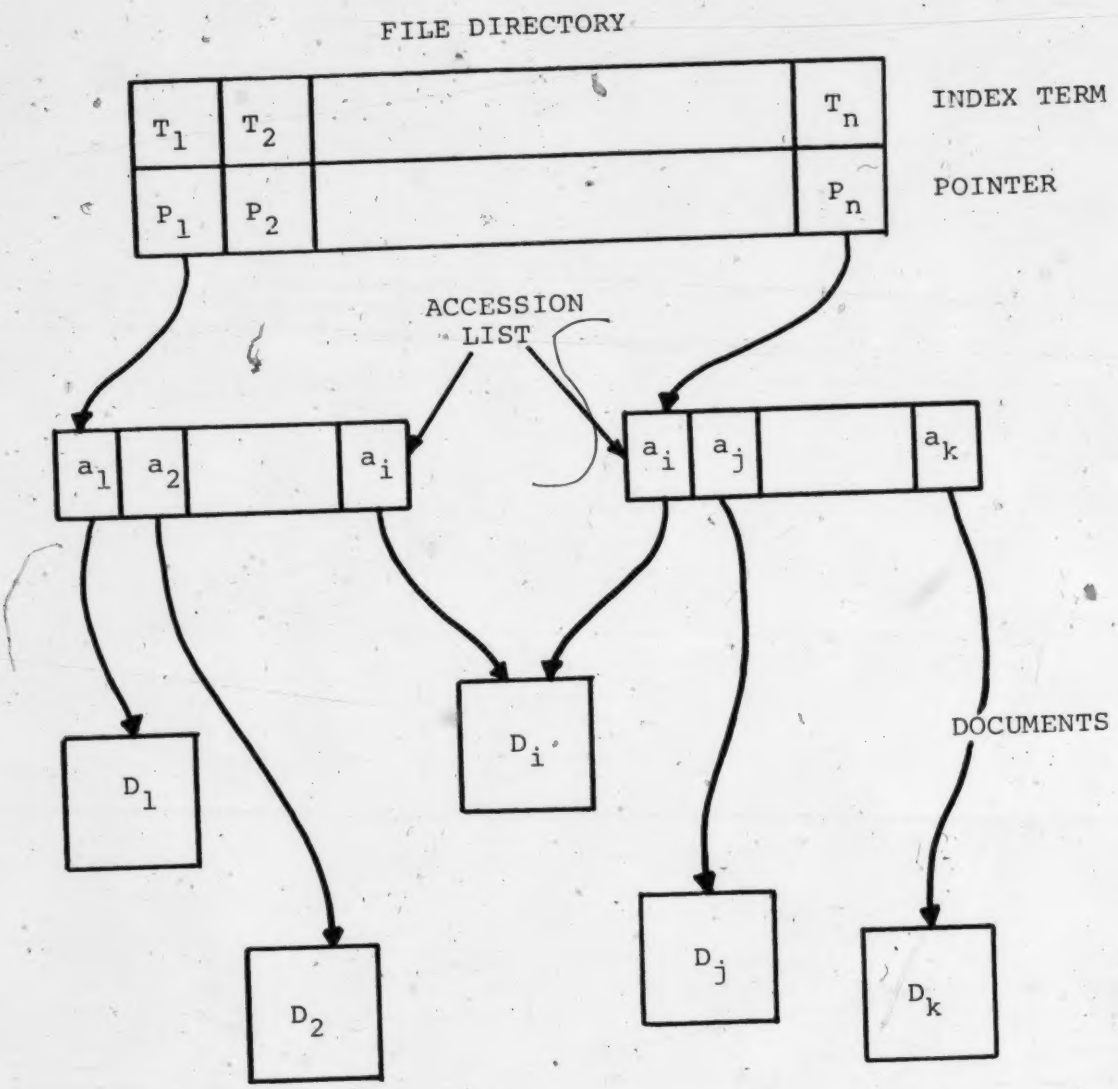


STRUCTURE OF A CHAINED FILE ORGANIZATION

FIGURE 1.1

C. Inverted Files

The design of an inverted file is essentially the same as that of a chained file, the only difference is that every index term of the directory in an inverted file is associated with a set of pointers which is known as the accession list. Figure 1.2 depicts the organization of an inverted file. For any given query, the search begins by scanning the directory to obtain the accession list for each query term. All the fetched lists are then merged to form a single list. Documents are then obtained by using the pointers in the merged list and correlated with the given user. Documents with 'high' correlations are then retrieved. During the process of list merging, the inverted file scheme eliminates many duplicate documents, thus fewer number of documents will be accessed from the secondary memory and that represents a substantial amount of retrieval time being saved. However, the inverted file scheme also introduces tremendous amount of storage overhead.



STRUCTURE OF AN INVERTED FILE ORGANIZATION

FIGURE 1.2

D. Clustered Files

All documents in the file are divided into subsets or clusters by using a certain classification procedure such that related documents are placed in the same cluster. Some examples of classification procedures are due to Needham, Doyle, Rhodes, Rocchio and others[3,4,5,6].

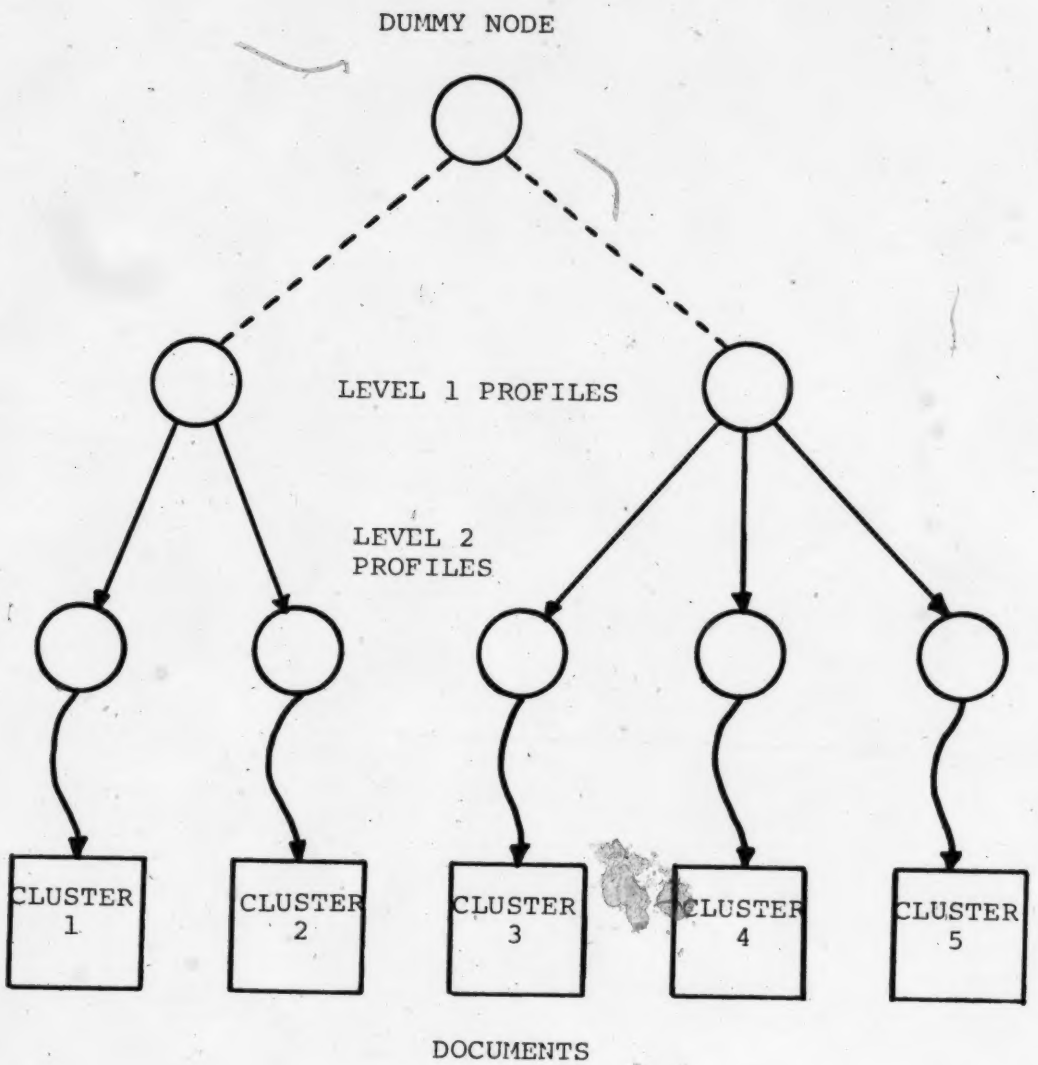
The contents of the documents in each group or cluster are then described into a profile or centroid. Some profile definitions are described in Murray[2]. A hierarchy of profiles can be built if the number of profiles is large. The user query is correlated with the profiles and the clusters which have a large number of related documents are retrieved. Figure 1.3 depicts the organization of a cluster file with 2 levels of profiles.

There is always a trade off between cost and performance in cluster files. More 'close' documents will be retrieved, if more clusters are searched and the cost will be higher. One major disadvantage of using cluster files is the expense in the initial classification procedure (clustering).

The maintenance of a cluster file is also a problem. Initially new documents can be inserted into the appropriate clusters without changes to the profiles. However, after a period of time, the quality of the profiles diminishes because the profiles try to represent too much information.

Eventually the whole file organization deteriorates and a reclassification has to be done.

Since computer systems retrieve one block of data at a time from the secondary storage and each node in the file (e.g. a profile or cluster) is usually packed in the same block, the access time is proportional to the total number of nodes searched. Furthermore, since actual documents are available to be examined directly, relevance feed back can easily be implemented.



STRUCTURE OF A CLUSTERED FILE WITH TWO LEVELS OF PROFILES

FIGURE 1.3

In summary, we have described four types of file organizations and each has a different design philosophy. Since the speed, storage requirement and flexibility play an important role in the success of a document retrieval system, comparisons among the four organizations in these three aspects are made as follows. First, in terms of speed, it is quite obvious that a sequential file is not capable of giving a fast response to the user query, since a sequential scan of the data file is required to locate the desired record. For chained files, in order to obtain the desired documents, a scan of the directory is required to obtain a list of documents for each query term and then a merging of all the documents together. Therefore the access time is proportional to the number of documents in all the lists associated with the query terms. The inverted file is designed to have a faster access time over the chained file and the retrieval time is proportional to the number of query terms and the number of documents retrieved. Note that the number of documents retrieved is usually a small fraction of the number of documents examined by a chained file organiztion. A clustered file can also provide a relatively fast response time to the user query as compared to that of the sequential and chained files, since it only involves the retrieval of the appropriate profiles and the corresponding clusters. Furthermore, a clustered file is

likely to have a faster retrieval time than the inverted file when the number of query terms increases, since its retrieval time is relatively independent of the number of query terms.

In terms of storage requirement, the amount of storage required by a sequential file is minimal, since no pointers, directory or other overhead are involved. In the case of a chained file and especially an inverted file, since a large directory has to be maintained, the storage overhead involved could make the file size rather large as well. In the cluster scheme, the overhead is the space used by the profiles. It is difficult to state the total storage requirement accurately since it depends on the number of hierarchy levels used. However according to Murray [2] the performance level can still be maintained even when the profiles are reduced to less than 10% of the space used by the documents.

Finally, in terms of flexibility, inverted and chained files can supply some pre-search statistic, for example, the number of documents containing a number of specified index terms. The clustered scheme cannot provide this same data, but offers the possibility of interrupting a search and viewing checkpoint information.

From the above comparisons, we can conclude that both inverted and clustered schemes are suitable for on-line processing. However in this thesis, we will concentrate on the retrieval problems of the cluster file organization.

1.3 RETRIEVAL IN CLUSTER FILES

In some cluster file retrieval systems, the profiles are generally much smaller than the clusters. Therefore they can be kept in the main memory for fast access while the clusters are stored in the secondary memory.

In large data sets, the number of clusters and profiles involved could be large thus it may be advantageous to group 'similar' profiles together to form a hierarchy of profiles. However for the sake of this presentation, we will assume one level of profiles, though the same approach can be applied to multi-level profiles.

In practice, the user specifies a query and N, the number of records to be retrieved. The retrieval system then correlates the query with the profiles and the clusters whose profiles have high correlations with the query are retrieved. The query is then correlated with the records in the retrieved clusters and the records are ranked according to the correlations. The N records with the highest correlations are then retrieved.

There are a variety of ways to correlate queries with profiles. This thesis essentially examines the method as proposed by C. T. Yu and W. S. Luk[10].

Their method is based on a probabilistic model such that prediction on the number of documents in a given range of 'accuracy' is made on every cluster. The definition of accuracy, according to their model, is the total number of attributes in common between a query and a document. Thus the clusters can be ranked according to the number of documents in a certain accuracy level. The clustering model is introduced in the next chapter.

CHAPTER 2

CLUSTER FILE RETRIEVAL SYSTEMS

2.1 A PROBABILISTIC MODEL

Most users of cluster file document retrieval systems are interested in obtaining a certain number of documents which are closest to their queries. For example, a cluster file information retrieval system for a real estate company can be designed to provide a list of houses which will match as many customer specified features as possible. Most cluster file document retrieval systems generally do not provide the users, before the retrieval of the actual records, with information such as estimates of the number of records which are reasonably close to the user queries and the clusters where these records can be located. However if this kind of information is available, then the users are able to adjust the query to obtain an optimum amount of output. For example, if there are too many records that are 'close' to the query, the user can reduce the number of retrieved records by imposing more constraints on the query.

This thesis investigates the applicability and possible improvements of a probabilistic model which is constructed according to the above-mentioned facts. This model includes all common features of a cluster file information system and in addition, it will give estimates of the number of

documents in each cluster at a certain degree of 'closeness' to a given user query. Suppose there are N attributes (a1,...,aN) in the entire data base and every record or document is defined to be an N-dimensional binary vector. The queries to the system are also defined in the same way. A '1' in the ith position or co-ordinate of a document or query indicates the presence of the ith attribute and a '0' indicates otherwise. The profile P of each cluster is defined to be the vector sum of all the records. Thus if P=(p1,...,pN), then pi is the number of records in the cluster having the ith attribute. Furthermore, the closesness between a query X=(x1,...,xN) and a document Y=(y1,...,yN) is measured by the number of terms or keywords in common between them. Formally, their closeness can be measured by the simple matching function $\sum_{i=1}^{N} X_i Y_i$

The analysis of this model is based on the following assumption:

(i) the attributes in the data base are mutually independent.

With the above assumption, the expected number of records in a cluster of M records, having exactly i attributes in common with a given query Q is derived as follows:

let Q be a query which contains exactly L attributes.

Without loss of generality, let Q contain the first L attributes in the data base.

Let the profile of the cluster be P=(p1,...,pN). The probability that a randomly chosen record from the cluster contains attribute ai, 1<=i<=L is pi/M. On the other hand, the probability that a record does not contain ai is (1-pi/M). Thus by the independence assumption, the probability that a record contains attributes a1,...,ai but not ai+1,...,aL is

$$\prod_{k=1}^{i} P_k/M \prod_{k=i+1}^{L} (1-P_k/M)$$

Since there are $\binom{L}{i}$ different ways of choosing i attributes out of L, the probability that a randomly chosen record has exactly i attributes in common with Q is equal to

$$\sum_{\binom{L}{i}} \left(\prod_{j=1}^{i} P_{g(j)}/M\right) \left(\prod_{j=i+1}^{L} (1-P_{g(j)}/M)\right) \qquad (2.1)$$

where { 1,...,L } is the disjoint union of {g(1),...,g(i) } and {g(i+1),...,g(L)}, g being a permutation and $\sum_{\binom{L}{i}}$ denotes the sum over all $\binom{L}{i}$ possible choices. We shall denote the combinatorial expression by $C(a_1,...,a_L,i)$ and call it the C-function. The expected number of records having K or more attributes in common with the query is $M \sum_{i=k}^{L} C(a_1,...,a_L,i)$.

By using the expressions that are derived here, it is possible to predict in the given cluster file the number of documents which are closely related to the user query. Thus clusters having high expected number of closely related

records can be selected for retrieval.

Even though the relevance of a retrieved document is not handled here, it is likely that the more attributes that a retrieved document has in common with a given query, the higher the chance the document is relevant to the query.

2.2 TO COMPUTE THE C FUNCTION

According to the definition, the C-function is not easy to compute. Since it involves a high polynomial or even exponential number of combinations of the frequencies of the attributes (as demonstrated in section 4.2), the time spent in computing the function for a given cluster may exceed that of retrieving all the records in the cluster. In this thesis, a more efficient method is used and is outlined as follows:

Lemma 2.1

Let Q=(q1,...,qL) be a query and (f1,...,fL)=(p1/M,...,pL/M) are the corresponding relative frequencies of occurrence in a cluster of M records.

$$\text{IF } F \equiv a_o + a_1x + a_2x^2 + \ldots + a_Lx^L$$
$$= (1 - f_1 + f_1x)(1 - f_2 + f_2x) \ldots (1 - f_L + f_Lx)$$
$$= (1 - f_1)(1 - f_2) \ldots (1 - f_L) + \ldots + (f_1f_2 \ldots f_L)x^L$$

then

ai is the probability that a document in a cluster with exactly i attributes in common with the query Q.

Proof

It is trivial for the case of a0 and aL. Since a0=(1-f1)...(1-fL), a0 is the probability that a document does not

contain q1,...,qL. Similarly the same reasoning can be applied to aL.

Consider the case of ai.

$$a_i = f_1 f_2 \cdots f_i (1 - f_{i+1})(1 - f_{i+2}) \cdots (1 - f_L) +$$
$$f_1 f_2 \cdots f_{i-1}(1 - f_i) f_{i+1}(1 - f_{i+2}) \cdots (1 - f_L) +$$
$$\vdots$$
$$(1 - f_1)(1 - f_2) \cdots (1 - f_{L-i}) f_{L-i+1} \cdots f_L$$

$$= \sum_{\binom{L}{i}} \left(\prod_{j=1}^{i} P_{g(j)}/M \right) \left(\prod_{j=i+1}^{L} (1 - P_{g(j)}/M) \right)$$

$$= C(q_1,\ldots,q_L,i)$$

=the probablity of a record has exactly i attributes in common with a given query.

Where { 1,...,L } is the disjoint union of the permutations $\{g(1),\ldots,g(i)\}$ and $\{g(i+1),\ldots,g(L)\}$

For the remainder of this thesis F is referred to as the generating function of a query Q. The following lemma is an extension of Lemma 2.1.

Lemma 2.2

Let Q1 and Q2 be two disjoint queries whose generating functions are G1 and G2 respectively. If the terms in Q1 are independent from those in Q2, then G = G1 * G2 is the generating function for Q1 U Q2, i.e. the probability that a document in a cluster has exactly K terms in common with Q1 U Q2 is the coefficient of X in G.

Proof:

Let $G1 = a_o + a_1 x + \ldots + a_i x^i + \ldots$

$G2 = b_o + b_1 x + \ldots + b_j x^j + \ldots$

$G = G1 * G2$

Consider the term

$$\sum_{\substack{0 \le i,j \le k \\ i+j=k}} (a_i b_j) x^{i+j} \quad \text{in } G \qquad 0 \le k \le L$$

since a_i is the probability that i terms in Q1 co-occur

and

b_j is the probability that j terms in Q2 co-occur

and

the terms in Q1 and that in Q2 are independent

$\sum_{\substack{0 \le i,j \le k \\ i+j=k}} a_i b_j$ is the probability that i+j=K terms in Q1 U Q2 co-occur

Thus G is the generating function of Q1 U Q2.

2.3 EXPERIMENT ON A REAL DATA SET

In the previous section, we have derived estimates on the number of records in a cluster having K or more attributes in common with a given query (the C-function) and the techniques to compute the estimates. In this section, an experiment is set up for applying the retrieval concept on an actual data set and comparing the results obtained with the theoretical estimates. The procedures in carrying out the experiment is outlined as follows:

(i) The CRN1400 collection is selected from the SMART RETRIEVAL SYSTEM [1] as the testing data set. The collection consists of 1400 documents and 225 queries.

(ii) Subroutine CLUSTER which uses the clustering algorithm designed by J. J. Rocchio [6] is selected from the SMART RETRIEVAL SYSTEM to perform the clustering on the documents of CRN1400 collection. Profiles are then created from the resulting 16 clusters.

(iii) The C-function of each query with respect to each

cluster is obtained. Documents are retrieved from the cluster whose C-function value is the highest.

(iv) Let Oki, Tki be the actual and expected number of documents with K or more attributes in common with the ith query. (In the remaining part of this thesis, K-value is referred to as K or more attributes in common with a given query).

Since Tki is a real number and Oki is an integer, the estimation does not seem to be realistic enough. In order to make the estimation more meaningful, an integer upper and lower bound are used. Thus if Oki is within the two bounds then the estimation error is zero. The upper and lower bound used in this thesis are $\lceil T_{ki} \rceil$ and $\lfloor T_{ki} \rfloor$

Where $\lfloor T_{ki} \rfloor$ is the greatest integer smaller than or equal to Tki and

$\lceil T_{ki} \rceil$ is the smallest integer bigger than or equal to Tki

The percentage error is then defined to be

$$E_{ki} = \min \left\{ \left| O_{ki} - \lceil T_{ki} \rceil \right| , \left| O_{ki} - \lfloor T_{ki} \rfloor \right| \right\} / O_{ki} * 100$$

The average error percentage over M1 queries on the estimation of the number of documents with K or more attributes in common with the queries is defined to be

$$\text{ave err} = \sum_{i=1}^{M1} E_{ki}/M1$$

The result of the experiment is shown in Table 1. Queries of approximately the same length are grouped together and the average error percentage at each K-value over all queries in a group is obtained. Even though in some cases the average error percentage is quite low (less than 10%), there are cases where the average error percentage is quite high. Hence an attempt is made to locate the source of errors. In the next chapter, an error analysis is made and the C-function is modified and better results are obtained.

TABLE 1

Experimental results by using CRN1400 collection
(225 queries and 1400 documents)
no query modification
k = the minimal number of attributes which a record possesses in common with a query

length of queries = 3 - 6

k	no. of queries	ave err
2	55	13.44
3	51	21.78
4	34	27.35
5	13	8.55
6	3	0.00

length of queries = 7 - 8

k	no. of queries	ave err
2	56	8.56
3	56	19.46
4	50	33.68
5	32	18.26
6	19	14.18
7	8	12.50
8	1	0.00

length of queries = 9 - 10

k	no. of queries	ave err
2	50	10.58

3	50	14.40
4	49	30.13
5	44	27.71
6	29	19.40
7	12	15.98
8	5	0.00
9	1	0.00
10	1	0.00

length of queries = 11 - 12

k	no. of queries	ave err
2	35	9.20
3	35	11.48
4	35	26.47
5	35	30.46
6	25	34.36
7	19	22.86
8	11	16.06
9	5	6.67
10	4	0.00
11	1	0.00

length of queries = 13 - 18

k	no. of queries	ave err
2	29	13.25
3	29	7.76
4	29	18.75
5	28	37.02
6	28	42.85

7	23	30.07
8	18	10.83
9	10	15.00
10	6	5.56
11	3	0.00
12	2	0.00
13	1	0.00

CHAPTER 3

ERROR ANALYSIS AND QUERY MODIFICATION

3.1 ERROR ANALYSIS

In observing the results presented by different queries from CRN1400, the theoretical values are in most cases below the actual values in high K-value (ref. Section 2.3 for definition). This simply means that some attributes co-occur more often than is assumed. In fact, the occurrence of the attributes of a given query are usually not mutually independent.

The other source of error is coming from the low estimation value at high K-value. Suppose the prediction or the C-function value is 1 and the actual value is 2, the error is already 50%, even though the difference between them is 1 only.

Despite the error caused by the latter source is quite high in some cases, we make no attempt to correct it, since the cause of the error is quite natural and is unrelated to the model of this thesis. However, an attempt is made to correct the former source by relaxing the attribute dependency assumption and an analysis is outlined in the next section.

3.2 ASSUMPTION RELAXATION

In an actual data base, it is usually not true that all the attributes are independent and in fact some of the attributes almost always occur together in the same documents and this is a sign of attributes dependency. The C-function is derived by assuming the independence of attributes and violation of this assumption would lead to the inaccuracy of the C-function. One way to remedy this defect is as follows: divide the attributes of a given query Q into two sets: set S1 and set S2. Where set S1 contains attributes from Q which are dependent on each other and set S2 contains attributes from Q which are independent of each other and independent from the attributes in S1. The theorectical generating function of S1 and S2 with repect to each cluster are then obtained. According to lemma 2.2, the theoretical generating function for Q can be expressed as a product of 2 polynomials.

$$F = (b_o + b_1 x + \ldots + b_i x^i)(c_o + c_1 x + \ldots + c_{L-i} x^{L-i})$$

suppose $f_1 = (b_o + b_1 x + \ldots + b_i x^i)$ is the theoretical generating function of the dependent attributes. It is quite reasonable to assume that this generating function is responsible for the big error percentage, since the attributes do not

satisfy the independent assumption of the model. Thus we can expect to improve the accuracy of F, if f1 is somehow modified. One way to modify f1 is by replacing it with the actual co-occurrence polynomial $f_2=(a_o+a_1x+ \ldots +a_ix^i)$. Note that the actual co-occurrence polynomial of a query is a polynomial whose coefficient ai represents the relative frequency of the number of documents in a given cluster with exactly i attributes in common with the given query. Thus after replacing f1 by f2, F now becomes

$$F = (a_o + a_1x + \ldots + a_ix^i)(c_o + c_1x + \ldots + c_{L-i}x^{L-i}).$$

In order to find out whether the query splitting concept can really improve the retrieval result, an experiment based on this concept is devised. Apparently, one problem in conducting this experiment is to determine in each cluster the dependency of the attributes. Unfortunately, this is a very expensive process, since all the co-occurrences of i different attributes for $1 <= i <= N$ and all possible combinations in each case have to be considered. In fact the time bound needed to complete the process is of $O(2^n)$, where n is the total number of attributes in the data base.

Assuming the data base has been in use for some time so that a set of representative user queries is obtained, the set of queries can then be used to determine the dependent relation of the most frequently occurred attributes (to be described below). The results will then be used to assist in the retrieval of documents by future queries. In this

thesis, an experiment is set up to determine the dependency relation of the most frequently occurred attributes and it is described as follows:

The set of user queries is divided into two sets: set A and set B, such that the attributes in any query in set B are a subset of the attributes of the queries in set A. Set A is then called the training set and set B, the testing set. The training set is used as a source for finding all the dependent attributes with respect to each cluster of the data base. The results are used to modify the generating functions of the queries in the testing set by using the query splitting concept. The same retrieval procedure as described in section 2.3 is then applied to the queries in the testing set and hopefully more accurate prediction can be obtained.

In a realistic retrieval environment, the data base is assumed to have been used for a reasonably long period of time. Thus representative queries can be collected. This set of queries can be used as set A in the above description. Recent and future queries then correspond to set B. Therefore, in practice, the strategy described above can be implemented.

3.3 THE DEPENDENT ATTRIBUTES

In this thesis, an algorithm which uses the contingency table (APPENDIX 1) is developed to locate the dependent attributes in a given query. The details of the algorithm are given as follows:

Definition: let $F_1=c_o+c_1x+ \ldots +c_Lx^L$ be the generating function of a given query with respect to a cluster and $F_2=a_o+a_1x+ \ldots +a_Lx^L$ is the actual relative frequencies of co-occurrence function with respect to the same cluster. Then $\max \left\{ \frac{|c_i-a_i|}{a_i}, 0 \le i \le L \right\}$ is defined to be the error of F1.

ALGORITHM 3.1

Let Q=(q1,...,qL) be the given query, E an error tolerance and C be the cluster from which the query retrieves documents. The error of the generating function of Q is first obtained. If the error is less than or equal to E then Q is assumed to have no dependent attributes and the algorithm stops. If the error is greater than E, then all possible combinations of two attributes are enumerated and for each pair of attributes, their chi-square value is

obtained from the contingency table. Each pair of attributes is then ranked in descending order of their chi-square value. Since the higher the chi-square value, the higher is the probability that a pair of attributes are dependent (APPENDIX 2), it is reasonable to assume that the highest ranked attribute pair are more dependent than the next highest ranked pair and so on. Thus, the generating function of Q is modified to be F = F1 * Fr, where F1 is the actual co-occurrence function of the highest ranked attribute pair and Fr is the theoretical generating function of the C-function of the rest of the attributes. The error of F is then computed. If it is smaller than E then it may be asserted that there is only one pair of attributes in Q. However, if the function error of F is still greater than E then F is modified further into F = F2 * Fr, where F2 includes the actual co-occurrence function for the first and the next highest ranked attribute pair and Fr is the theoretical generating function for the rest of the attributes. This procedure is repeated until the error is less than E. The details of this algorithm is given in APPENDIX 3.

3.4 EXPERIMENT SETUP AND RESULTS

The experimental environment remains the same as that described in section 2.3. Instead of using all 225 queries in CRN1400 for retrieving documents, they are divided into two sets, set A(155 queries) and set B(70 queries). (ref. Section 3.2 for definition). Essentially set A is used to obtain dependent attributes with respect to each cluster in the data base. The dependent attributes are then used by set B to modify the C-functions of its queries for the retrieval of documents.

The dependent attribute set Di, with respect to cluster i, is obtained by applying algorithm 3.1 to each query in set A. The attributes of each query in B with respect to each cluster are then divided into two sets: the dependent and the independent attribute set. The dependent set in a given query in B with respect to cluster i is obtained from the intersection of the query and Di. The independent set consists of the remaining attributes of the query. The generating function for each query with respect to cluster i is then of the form F = F1 * F2, where F1 is the actual frequencies of co-occurrence function of the dependent attributes with respect to cluster i and F2 is the

generating function for the independent attributes. Documents are then retrieved from the clustered data base (as described in section 2.3). The results as shown in TABLE 2 (with an average error of 11.06%) indicate improvement over the original results (with an average error of 13.12%). The following is a diagram showing the performance of a typical query.

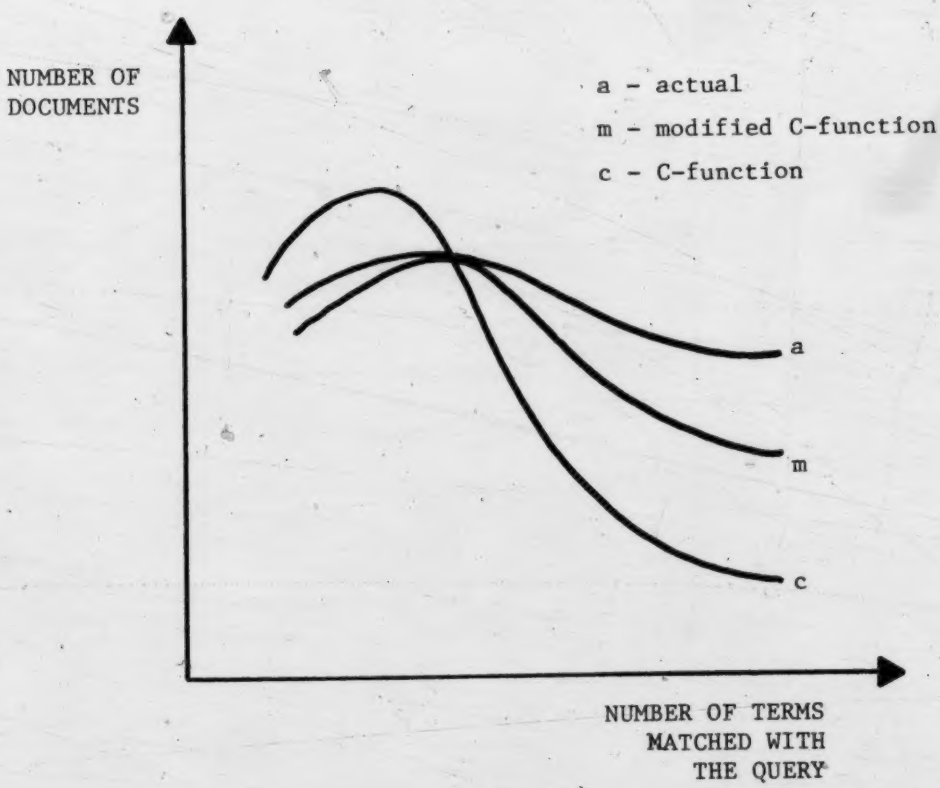


DIAGRAM 3.1

THE PERFORMANCE OF A TYPICAL QUERY IN SET B

BY USING THE C-FUNCTION AND THE MODIFIED C-FUNCTION

TABLE 2

Experimental results by using set B queries only. The C-function was modified by using the dependent term set Di.

k = the minimal number of attributes which a record possesses in common with a query

length of queries = 3 - 5

k	no. of queries	ave err
2	19	4.94
3	16	5.46
4	10	25.00
5	5	0.00

length of queries = 6 - 7

k	no. of queries	ave err
2	17	3.45
3	17	6.51
4	17	8.18
5	10	10.00
6	7	7.14

length of queries = 8 - 9

k	no. of queries	ave err
2	22	4.07
3	22	3.31
4	22	7.39
5	20	18.16

6	17	10.59
7	6	0.00
8	2	0.00

length of queries = 10 - 15

k	no. of queries	ave err
2	12	5.91
3	12	6.40
4	12	9.06
5	12	23.60
6	10	22.71
7	8	24.41
8	6	12.50
9	2	37.50
10	2	0.00
11	1	0.00

From the performance of the queries in set B, the following observations are made:

(i) When the number of dependent attributes in a query is less than L/2 then the error increases approximately at k > L/2, where L is the length of a query.

(ii) In most cases the modified C-function is underestimating the true value at k > L/2, implying there are still signs of dependency.

It was believed that more improvement can be achieved by modifying the C-function further with the above-mentioned observations. The observations suggest that the independent attributes in a query are also dependent to a certain extent, even though their dependency is not as strong as that of the independent attributes. Thus by using the error made by the dependent attributes as an upper bound (the difference between the C-function and the modified C-function), we can modify the C-function further by shifting the modified C-function value towards the actual value. The following is a modification scheme used by this thesis.

Let the query Q=(q1,...,qL) and the corresponding frequency be (p1,...,pL)

(i) the modified C-function is modified further at L/2.

Denote the difference between the modified C-function and the C-function at K > L/2 by $MC_{L/2}$.

(ii) without loss of generality, let D1=(q1,...,qi) be the dependent attributes of Q and D2=(qi+1,...,qL) be the remaining set.

(iii) if i >= L/2 then no further modification is required.

(iv) if i < L/2 then let $F_1 = \sum_{j=1}^{i} P_j/i$ and $F_2 = \sum_{j=i+1}^{L} P_j/(L-i)$

(v)

$$MC'_{L/2} = MC_{L/2} * (1 + \sum_{j=i+1}^{L} P_j \; / \; \sum_{j=1}^{i} P_j)$$

where $MC'_{L/2}$ is the difference between the new modified C-function and the C-function. This step 'shifts' the modified C-function value(as shown in diagram 3.1) towards the acutal value.

The modification is based on the assumption that the attributes in D2 are also dependent to a certain extent(i.e. we also have to consider the error made by the remaining L-i attributes). Since the dependency of the attributes in D2 is not as strong as that in D1, $MC_{L/2}$ can be considered as an upper bound to the error made by any i out of L attributes. Thus it is reasonable to assume that the maximum amount of error made by the remaining L-i attributes made is $MC_{L/2} * (L-i)/i$.

Furthermore the document frequencies of the attributes also affect the modification. Suppose the average document frequency of D1 is 50 and the error made by these attributes

is 20, then we expect that the error made by D2 with an average document frequencies of 10 is less than 20. In fact this thesis assumes that the error made by D2 is proportional to the document frequencies. Thus the new modified C-function value becomes

$$MC_{L/2} * \frac{L-1}{i} * \frac{F_2}{F_1}$$

Therefore the difference between the new modified C-function and the C-function is

$$MC'_{L/2} = MC_{L/2} + MC_{L/2} * \frac{L-i}{i} * \frac{F_2}{F_1}$$

The refined modification procedure was then applied to set B queries. The results as shown in TABLE 3, indicate an improvement in all cases. The average error is about 6%. For the group of queries of length 10 - 15, significant improvement is obtained.

TABLE 3

Experimental results by using set B queries only. The C-function was modified by using the dependent term set D and the refined modification procedure.

k = the minimal number of attributes which a record possesses in common with a query

	length of queries = 3 - 5	
k	no. of queries	ave err
2	19	4.94
3	16	2.34
4	10	15.00
5	5	0.00
	length of queries = 6 - 7	
k	no. of queries	ave err
2	17	3.45
3	17	6.36
4	17	7.13
5	10	10.00
6	7	0.00
	length of queries = 8 - 9	
k	no. of queries	ave err
2	22	4.07
3	22	3.31
4	22	6.98

5	20	11.91
6	17	6.47
7	6	0.00
8	2	0.00

length of queries = 10 - 15

k	no. of queries	ave err
2	12	5.91
3	12	6.40
4	12	9.06
5	12	19.76
6	10	10.56
7	8	15.38
8	6	8.33
9	2	12.50
10	2	0.00
11	1	0.00

CHAPTER 4

TIME BOUND ANALYSIS OF ALGORITHMS

4.1 INTRODUCTION

In order to choose an appropriate algorithm from a set of potential algorithms which can be used as a solution to a problem, it is desirable to evaluate the cost of each. The most common criteria used to evaluate algorithms are the time and space consumed.

Essentially, these two criteria are measured in terms of the 'size' of a problem which can be defined as a measure of the input quantity[9]. For example, in adding two matrices, the size of the problem is the dimension of the matrices. In computing the C-function, the size of the problem is the length of the query.

The computing time consumed by an algorithm can generally be expressed as a function of the size of the problem. The time complexity of an algorithm is said to be of order f(n), if there exists a constant c such that the number of steps executed by the algorithm is always less than or equal to cf(n), where n is the size of the problem. For example, an algorithm takes $2n^3$ operations to compute has time complexity of order n^3(or denoted as $O(n^3)$). A similar definition can be applied to the space consumed.

Furthermore, according to their time complexity,

algorithms can roughly be classified into two groups: deterministic polynomial and exponential.

Definition. An algorithm is said to be deterministic polynomial if there exists a polynomial p(n), such that the number of steps executed by the algorithm is less than or equal to p(n), where n is the size of the problem.

Definition. An algorithm is said to be exponential if it runs in exponential time, i.e. the algorithm is of order n^k, where $k > 1$ is a constant and n is the size of the problem.

Exponential algorithms are usually applied to problems of smaller size, since the amount of running time required by an exponential algorithm for a large size problem will be too large for the algorithm to be feasible.

On the other hand, deterministic polynomial algorithms of low degrees are generally desirable for most applications. The algorithms used in computing the C-function, for example, are deterministic polynomial of degree 2 (will be shown in section 4.3).

The analysis of time bound required by some algorithms used in this thesis is described in the following sections.

4.2 TIME BOUND FOR COMPUTING THE C-FUNTION ACCORDING TO ITS DEFINITION

let the given query be Q=(q1,...,qL) and its relative frequencies of occurrence with respect to a cluster C of M records be F=(f1,...,fL).

Recall that C-function is defined to be the probability that a record in C contains exactly i out of L attributes in Q.

From equation 2.1, C-function can be expressed as follows:

$$C(q_1,q_2,\ldots,q_L,i) = \sum_{\binom{L}{i}} \prod_{j=1}^{i} f_{g(j)} \prod_{j=i+1}^{L} (1 - f_{g(j)})$$

Then the number of multiplications needed to compute

$$\prod_{j=1}^{i} f_{g(j)} \prod_{j=i+1}^{L} (1 - f_{g(j)}) \quad \text{is} \quad (L - 1)$$

Therefore the number of operations needed to compute

$C(q_1,q_2,\ldots,q_L,i)$ is $\binom{L}{i} * (L - 1)$

Since there are $\binom{L}{i}$ different ways of choosing i out of L attributes

Thus the total number of operations needed to compute

$$\sum_{i=0}^{L} C(q_1,q_2,\ldots,q_L,i) \quad \text{is}$$

$$\left\{\binom{L}{0} + \binom{L}{1} + \ldots + \binom{L}{L}\right\} * (L - 1) = 2^L(L - 1)$$

the time bound required to compute $\sum_{i=0}^{L} C(q_1,\ldots,q_L,i)$ is

therefore $O(L2^L)$. Since this algorithm runs in exponential time, it cannot be applied in general (especially in the case where the length of the query is long). Thus alternate algorithms are developed to compute the C-function.

The time bounds required by these algorithms are described in the next section.

4.3 TIME BOUND FOR POLYNOMIAL MULTIPLICATIONS

According to Lemma 2.1, the problem of computing the C-function is equivalent to obtaining the product of L polynomials of degree 1, where L is the length of a given query. There are many algorithms in solving this polynomial multiplication problem and the fastest known algorithm is called the Fast Fourier Transform[9]. The time bound required by this algorithm is O(L log L) with a reasonably large constant factor. As will be demonstrated later in this section, the usual way of multiplying L polynomials together is of $O(L^2)$ operations with a small constant. Thus, the Fast Fourier Transform is preferable to the standard method only if L is sufficiently large. However, the number of attributes in a query is usually not large enough to warrant the Fast Fourier Transform method. Two other algorithms with a slightly higher time bound are considered instead. The time bounds of these two algorithms are outlined as follows:

i) Iterative multiplication of L polynomials of degree 1.

Let y = (a1x+b1) * (a2x+b2) ...*(aLx+bL)

the numbers of multiplications and additions required in multiplying the first two polynomials are respectively 2 * 2 = 4 and 1.

The numbers of multiplications and additions required in multiplying the product of the first two polynomials by the third polynomial are 2 * 3 = 6 and 2 respectively.
In general the total number of operations needed is

$$(2*2+1) + (2*3+2) + \ldots + (2*L+L-1)$$

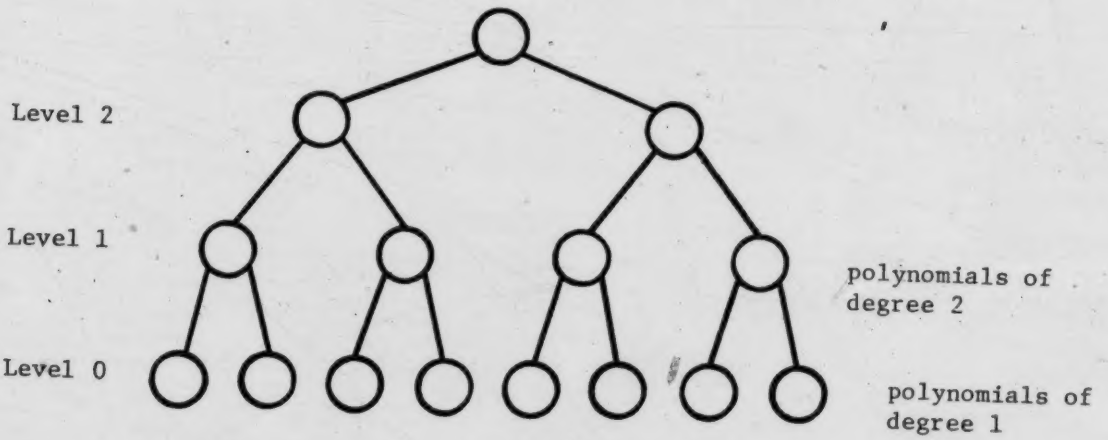
$$= 2 \sum_{i=2}^{L} i + \sum_{i=1}^{L-1} i$$

$$= \frac{(3L + 4)(L - 1)}{2} = 3/2\ L^2 + 1/2\ L - 2$$

thus the time bound required by this algorithm is $O(L^2)$.

ii) Group multiplication of L polynomials of degree 1. Assuming the initial number of polynomials is $L = 2^k$, where k is an integer and each polynomial is of degree 1. We can form a binary tree by using these L polynomials such that the nodes of the lowest level of the tree respresent the polynomials of degree 1. The parent of any two nodes represents the product of the two polynomials. The root of the tree is then the final product of all the terminal nodes.

A binary tree representing the multiplication of 8 polynomials of degree 1 is described below:



The number of additions needed to multiply two polynomials of degree 1 is $2 \sum_{i=0}^{0} i + 2^0 = 1$

In general the number of additions needed to multiply two polynomials of degree 2^j is $2 \sum_{i=1}^{2^j-1} i + 2^j$

The number of multiplications needed is $(2^j + 1)^2$ since the number of coefficients in a polynomial of degree 2^j is $2^j + 1$.

The total number of operations needed to multiply two polynomials of degree 2^j is $(2 \sum_{i=0}^{2^j-1} i + 2^j + (2^j + 1)^2)$

At each level there are $\frac{L}{2^{j+1}}$ groups to be formed where j is the level.

The total number of operations needed to go from level j to j+1 is

$$(2 \sum_{i=0}^{2^{j-1}} i + 2^j + (2^j + 1)^2)\frac{L}{2^{j+1}}$$

The total number of operations needed to build the whole tree is

$$\sum_{j=0}^{k-1} \left[2 \sum_{i=0}^{2^j-1} i + 2^j + (2^j + 1)^2 \right] \frac{L}{2^{j+1}}$$

$$= \frac{L}{2} \sum_{j=0}^{k-1} (2^{j+1} + 2 + \frac{1}{2^j})$$

$$= L^2 + L \, Ln(L) - 1$$

Thus the time bound required by the algorithm is $O(L^2)$.

Since algorithm (ii) has a smaller constant factor, it is used in this thesis for computing the C-function.

REFERENCES

[1] Salton, G., THE SMART RETRIEVAL SYSTEM - Experiment in automatic document processing, Prentice-Hall Inc., Englewood Cliffs, N.J., 1971.

[2] D. M. Murray, Information storage and retrieval. document retrieval based on clustered files, Cornell University doctoral thesis, report no. ISR-20, Cornell University, Ithaca, New York, 1972.

[3] R. Needham, The place of automatic classification in information retrieval, report ML-166, Cambridge Language Research Unit, Cambridge, England, 1963.

[4] L. Doyle, Breaking the cost barrier in automatic classification, SDC paper SP-2516, July, 1966.

[5] A. F. Parker-Rhodes, R. M. Needham, The theory of clumps, report ML-162, Cambridge Language Research Unit, Cambridge, England, 1960.

[6] J. J. Rocchio, Document retrieval system -- optimization and evaluation, Harvard University doctoral thesis, report ISR-10, Harvard Computation Laboratory, March 1966.

[7] D. Lefkovitz, File structure for on-line systems. Spartan Books, New York, 1969.

[8] R. M. Hayes, Information retrieval: an introduction Datamation, March 1968.

[9] Aho, A. V., Hopcroft, J. E. and Ullman, J. D., The design and analysis of computer algorithms, Addison-Wesley Publishing Comp., 1974.

[10] Yu C. T. and Luk W. S., Analysis of effectiveness of retrieval in cluster files, J. ACM(to appear).

[11] J. R. Files, H. D. Husky, An information retrieval system based on superimposed coding, Proceedings FJCC,

1969.

[12] Gene V. Glass and Julian C. Stanley, Statistical methods in education and psychology, Prentice-Hall, 1970.

APPENDIX 1

Definition: ϕ is defined to be the Pearson product-moment coefficient calculated on nominal-dichotomous data. Its main function is to measure the correlation between random variables. i.e. the higher the ϕ value between 2 variables, the higher is the correlation between them.

Gene V. Glass and Julian C. Stanley[12] state that: if Z is the standard normal random variable with mean 0 and variance 1 then

$$\sqrt{n}\,\phi = z \longrightarrow n\phi^2 = z^2$$

$$\phi^2 = z^2/n$$

since $z^2 = \lambda_1^2$ (Chi-square with 1 degree of freedom)

$$\phi^2 = \frac{\lambda_1^2}{n}$$

$\Longrightarrow$ the higher the chi-square value, the higher is the correlation between 2 variables.

APPENDIX 2

Let x11 be the number of documents in the data base that contain attributes X1 and X2.

Let x12 be the number of documents in the data base that contain attribute X2 but not X1.

Let x21 be the number of documents in the data base that contain attribute X1 but not X2.

Let x22 be the number of documents in the data base that do not contain attributes X1 and X2.

The following is a contingency table for attribute X1 and X2.

	$X1$	$\overline{X1}$	
$\overline{X2}$	X_{11}	X_{12}	W_1
$\overline{X2}$	X_{21}	X_{22}	W_2
	S_1	S_2	

Where $W_i = \dfrac{x_{i1} + x_{i2}}{2}$

$$S_j = \frac{x_{1j} + x_{2j}}{2}$$

$$n = \sum_i \sum_j x_{ij}$$

and $CHI = \sum_i \sum_j \dfrac{(x_{ij} - nW_iS_j)^2}{nW_iS_j}$ is called the chi-square value of X1 and X2.

APPENDIX 3

Let Q = (q1,...,qL) be a given query and E be the error tolerance and let V=(v1,...,$v_{\binom{L}{2}}$),
U = (u11,u12),(u21,u22),...,($u_{\binom{L}{2}-1}$,$u_{\binom{L}{2}}$2)) and
GF = (F1,...,$F_{\binom{L}{2}}$) such that v1>=v2>=...$v_{\binom{L}{2}}$ where vi and Fi are respectively the chi-square value(obtained from the contingency table) and the generating function of the attribute pair(ui1,ui2), 1<=i<=$\binom{L}{2}$, with respect to a cluster.

The procedure in carrying out ALGORITHM 3.1 is outlined as follows:

1) if the function error of the generating function of the C-function of Q is less than E, then all the terms in Q are independent and the algorithm stops.

2) let i = 1

3) let the generating function of the C-function for Q be
$F = \prod_{k=1}^{i} Fk * Fr$
where Fr is the generating function of the C-function of the set ($Q - \bigcup_{j=1}^{i}$ (uj1,uj2)). Note that if F = F1 * F2 * Fr and (u11,u12) $\cap$ (u21,u22) is not empty and suppose u12 = u21 then (u11,u12,u22) are considered as 3 dependent attributes

and F = F1 * Fr, where F1 is the actual co-occurrence function of (u11,u12,u22) and Fr is the generating function of the C-function for the set (Q - (u11,u12,u22)).

4) obtain the function error of F, if F is less than E then the dependent pairs are (uj1,uj2) j=1,i and the algorithm stops otherwise let i = i + 1 and go to step 3.